GOING SOCIAL

GEOSOCIAL NETWORKING SECURITY AND FACEBOOK SEMANTIC ANALYSIS

FACEBOOK SEMANTIC ANALYSIS

- That guy who knows everyone. **What's his trick**?
- Whom should we be **talking to less**, because **they're just so negative**?
- And who **just can't keep their mouth shut**, especially when it comes to **naughty words**?

Facebook Semantic Analysis can reveal these high level details from your messages archive.

Friend Class	
Name	String
Status	Byte (Enum)
Messages	List of Message

Message Class	
Contents	String
TheOtherPerson	Boolean
DateTime	String

"As a Subject has many Friends, each Friend has many Messages"

GEOSOCIAL NETWORKING SECURITY

- Social networking applications like Tinder and Grindr have **contextualised dating** and meeting new people through **location awareness**
- Location and immediateness have taken priority such that **compatibility and interests are now secondary** factors
- For security and safety, geosocial networking applications protect users' privacy by revealing *distance* instead of *exact location*…
- But this research **exploits Grindr** to prove insecurities even by nature.

EXPLOITING GRINDR

STEP 1 Run Grindr in a virtual Android machine so its location can be changed. This meant I didn't have to walk around the world with a phone…
STEP 2 Listen to the debug bridge, over which Grindr reveals its inner workings - especially dumps of users' profile information. People in whom I was interested I added as 'Favourites' for easy tracking (stalking) - Grindr processed these separately.
STEP 3 Parse the debug information, revealed over command line, and form a data structure for each person, logging inside their distance from each mock location at which the virtual machine sits.
STEP 4 Change the mock location automatically and triangulate to find exact locations within each data structure.

This led to **Stalkr**…

ANALYSIS METHOD

Frazzlr was created to allow everyday users to analyse their own Facebook data. This is how it works:

STEP 1

The program **takes a copy of your personal data archive** (downloadable from Facebook), which includes all messages you have ever sent and received. They are contiguously stored in a single huge HTML file:

```
<div class="contents"><h1>James Garner</h1><div class="thread">Adam Smith, James Garner<div
class="message"><div class="message_header"><span class="user">James Garner</span><span class="meta">Friday,
23 November 2012 at 08:56 UTC</span></div></div><p>Are you awake?</p><div class="message"><div
class="message_header"><span class="user">Adam Smith</span><span class="meta">Friday, 23 November 2012 at
10:01 UTC</span></div></div><p>Sorry about that, slept right through my alarm :/</p><div class="message"><div
class="message_header"><span class="user">James Garner</span><span class="meta">Friday, 23 November 2012 at
10:02 UTC</span></div></div><p>likewise</p> ...
```

STEP 2

The program **utilises a complex parsing algorithm** to store data from the archive in classes, such as *Friend* and *Message*. The archive also includes data about advertising targeting which isn't available via the Facebook website, so this data is stored in classes for analysis too, along side event attendance and the friends list.

These classes are linked such that a *Subject* class contains a list of *Friend* classes, each of which possesses a list of *Message* classes, and so on…

STEP 3

The program **includes sample basic tests**, but you can write your own simple tests in LINQ or by logic (for example, looping through each of a *Subject*'s *Friends* and totalling the *Messages* count, to judge a level of Facebook activity).

More complex tests generate tab-separated value files (plain text spreadsheets) for further sorting, analysis and calculations in Microsoft Excel. Then it's easy to generate charts!

```
Dim RunningTotal As UInt64 = 0
For Each ThisFriend In Whom.Friends
    RunningTotal += ThisFriend.Messages.Count
Next
OutputData = "You have sent and received " + RunningTotal.ToString + " messages."
```

Stalkr was created to automate parsing the debug information. The program interacts with the virtual machine and presents results visually.

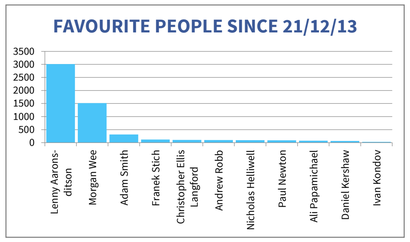
Triangulation with 2 radii. The green dot is the user's actual location. This was done from about 5500km away.

ANALYSIS RESULTS

'FAVOURITE PEOPLE SINCE 21/12/13'

Friend versus Message Frequency

For each person, the count is taken of each message where its timestamp exceeds 21/12/2013. This query is written in LINQ but could be run by traditional method through looping constructs.

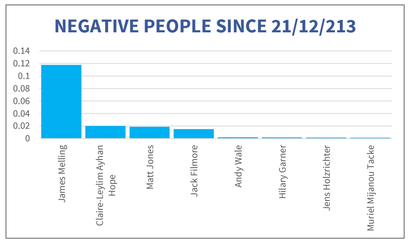


LIMITATIONS

- Weighted by time, when one person may be of particular interest (ideally, it should be weighted by the time between the first and the last message, yet this still does not account for long periods of communication absence)
- No consideration of message length: many essay-length messages should hold greater significance than '*kk*', '*gdgd*', '*ta*', '*ohh i see*'

FURTHER INVESTIGATIONS

- Correlation with 'perceived popularity': calculate the correlation co-efficient of message count ranking versus popularity ranking by survey
- Correlation with mathematical 'social models,' or the Bell curve and Exponential function



LIMITATIONS

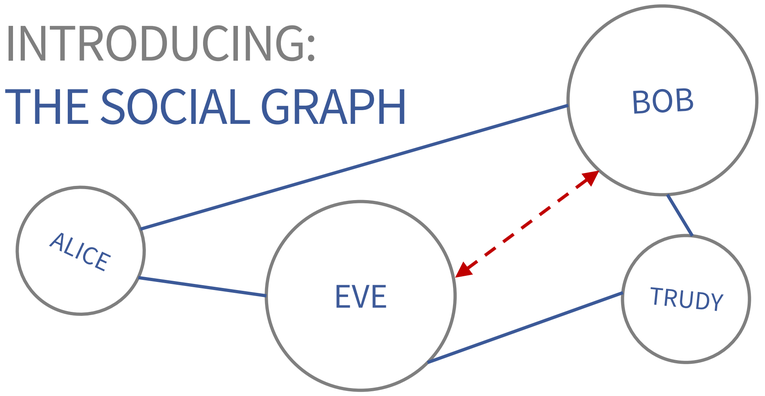
- No 'wider semantic analysis': positivity and negativity are better derived from text semantics than simple character analysis
- Of the three most insignificant Negativity Magnitudes, two people are German; this shows the test's simplistic language dependence

'NEGATIVE PEOPLE SINCE 21/12/13'

Friend versus 'Negativity Magnitude' (0-1)

For each person, the count of a static list of 'negative words' ('*grr*', '*annoyed*', '*hate*', '*stupid*', '*tw*t*') occurring is taken. This count is weighted by the total message count for the person. The most 3 significant and insignificant Negativity Magnitudes are graphed right.

INTRODUCING: THE SOCIAL GRAPH



"The social graph in the Internet context is a graph that depicts personal relations of Internet users"
Facebook: One Social Graph to Rule Them All?, July 11, 2010

THE NEXT STEP: CONSTRUCT A SOCIAL GRAPH

Frazzlr's next task is to create a social graph from messages and events

A simple Social Graph consists of interconnecting nodes (people) which are connected by relations. This Social Graph is mathematically constructed such that it can be rigorously analysed. For example, it is clear that node Alice is related to node Bob. It cannot be determined that node Alice is related to node Trudy, yet it is likely because they share the mutual nodes Eve and Bob. Conversely, an analysis of the Social Graph suggests a relationship between node Eve and node Bob because of their two shared mutual nodes. Hurrah! Friend Suggestions, anyone?

JAMES GARNER • 2ND YEAR COMPUTER SCIENCE • LANCASTER UNIVERSITY

InfoLab21 SHOWCASE